

Adventures in Modeling Spaces: Close Encounters of the Semantic Web and MDA Kinds

Dragan Djurić¹, Dragan Gašević² and Vladan Devedžić¹

¹FON – School of Business Administration, University of Belgrade
Jove Ilića 154, 11000 Belgrade, Serbia and Montenegro
dragandj@gmail.com, devedzic@fon.bg.ac.yu

School of Interactive Arts and Technology, Simon Fraser University Surrey
13450 102 Ave., Surrey BC V3T 5X3, Canada
dgasevic@sfu.ca

Abstract. This paper presents the original idea of Modeling Spaces and its impact on understanding various modeling domains and their interconnections. The focus is given on the Semantic Web and Model-Driven Architecture technical spaces and modeling spaces they are built around – RDF(S) and MOF modeling spaces. It also clarifies their mutual relations and helps understand what is necessary to do to bridge them and acquire their interoperability.

1 Introduction

Recent software engineering efforts rely on many concepts like models, metamodels, model transformations, etc. Although most of them try to exemplify most important benefits, they very often do not consider how software practitioners understand modeling. In fact, when talk about models, software engineers often think of a specific kind of models – UML models. However, there are many open questions such as whether we should assume the code we write as a model or not; what are models and metamodels and why do we need them; what means transforming a model into a programming language, etc.

In the past few years, two major approaches to modeling have emerged object-oriented (OO) Model-Driven Architecture (MDA) and ontology-based Semantic Web (SW). In this paper, we show how these competitive approaches can be generalized to the point where their mutual characteristics help us to explain how they can interoperate. Moreover, we present the original idea of Modeling Spaces (MS) [7], a formal framework for the understanding of different modeling approaches, which we then use to explain MDA and SW in a similar way and to identify how their connection can be acquired.

After the introduction, we present a short overview of SW and MDA. Then, we dedicate the third section to the definition and explanation of MSs, as they are new, original, concept. In the fourth section, we give the details of how MS can be used to explain the connection between RDF(S) (ontologies) and MOF (OO). The fifth chapter brings a more precise definition of technical spaces (TS) [11] with the help of

MS and the explanation of the connection between SW and MDA TSs. Then, we bring the conclusions and say something about the future work.

2 A short overview of the Semantic Web and Model Driven Architecture

The step beyond the World Wide Web is the Semantic Web [3], which enables machine-understandable data to be shared across the Net. The Semantic Web is powered by metadata, described by ontologies that give machine-understandable meaning to its data. Ontology is one of the most important concepts in knowledge representation. It can be generally defined as shared formal conceptualization of particular domain [8]. The World Wide Web and XML provide the ontologies with interoperability, and these interoperable ontologies, in return, facilitate Web that can “know” something.

Semantic Web architecture is a functional, non-fixed architecture [2]. Barnes-Lee defined three distinct levels that incrementally introduce expressive primitives: metadata layer, schema layer and logical layer [3]. Languages that support this architecture and the place of OWL are shown in Figure 1.

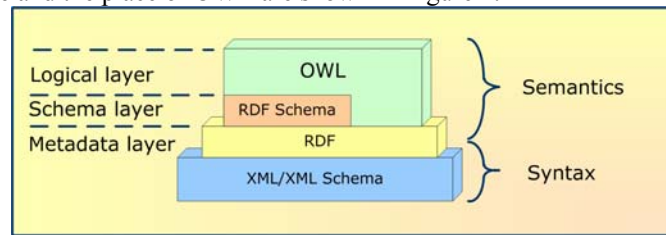


Fig. 1. OWL in the Semantic Web architecture

Common data interoperability in present applications is best achieved by using XML. As shown in the Figure 1, XML supports syntax, while semantics is provided by RDF, RDF Schema and mainly by OWL [9]. In order to provide capabilities for unconstrained representation of the Web knowledge and, in the same time, to support calculations and reasoning in finite time with tools that can be built on existing or soon available technologies, OWL introduces three increasingly expressive sublanguages for various purposes: OWL Full (maximal expressiveness), OWL DL (guaranties computational completeness) and OWL Lite (for starters).

Model Driven Architecture (MDA) defines three viewpoints (levels of abstraction) from which a certain system can be analyzed. Starting from a specific viewpoint, we can define the system representation (viewpoint model). The representations/models/viewpoints are Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM) [12]. MDA is based on a four-layer metamodeling architecture and several complementing OMG standards, Figure 2. These standards are Meta-Object Facility (MOF) [MOF 2003], Unified Modeling Language (UML) [14] and XML Metadata Interchange (XMI) [12],

and the layers are: meta-metamodel layer (M3), metamodel layer (M2), model layer (M1), and the real world layer (M0).

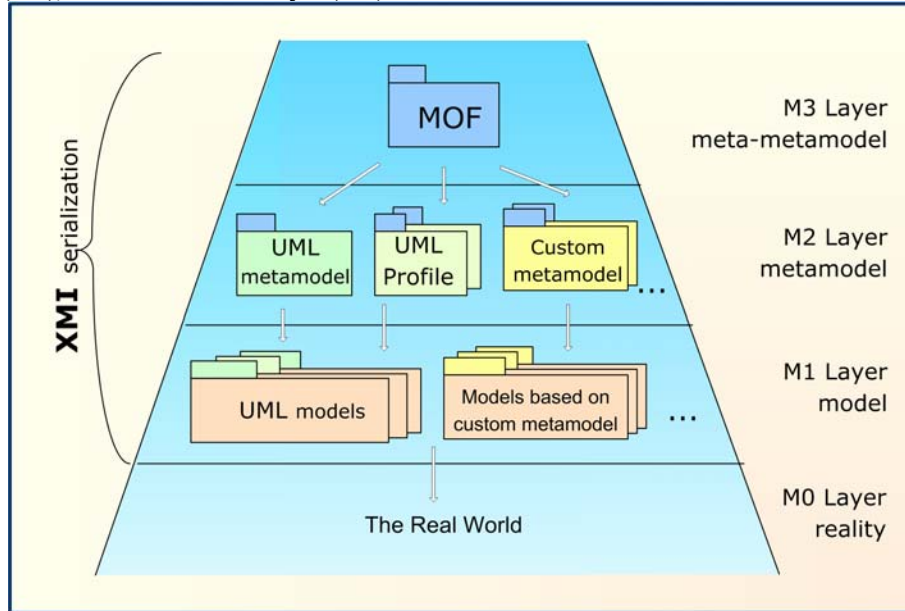


Fig. 2. MDA four-layer MOF-based metadata architecture

The topmost layer in this architecture (meta-metamodel, MOF) defines an abstract language and framework for specifying, constructing and managing technology-neutral metamodels. It is the foundation for defining any modeling language, such as UML or even MOF itself. All metamodels (both standard and custom) defined by MOF are positioned at the M2 layer. The models of the real world, represented by concepts defined in the corresponding metamodel at the M2 layer (e.g. UML metamodel) are at the M1 layer. Finally, at the M0 layer are things from the real world. The purpose of the four layers with common meta-metamodel is to support multiple metamodels and models and their scaling – to enable their extensibility, integration and generic model and metamodel management.

3 Modeling spaces essentials

The most fundamental definition of model is that it is the simplified abstraction of reality [15]. This definition applies not only to models in technology, but in art or everyday life. Having this definition in mind, we can draw two important conclusions. First, something can be taken as a model if it is an abstraction of things from the real world, but it is simultaneously a thing from the real world. Whether we take it as a model or as a real/world thing depends on the context, i.e. on the point of view. Second, models can be defined using metamodeling concepts formally or implicitly.

Since implicit metamodels cannot be precisely defined using formalisms, as in the case of art, in the rest of this discussion we analyze only formal models. Nevertheless, much of the conclusions can also be applied to implicit metamodels.

Figure 3 shows a general modeling architecture that was inspired by MDA and is in fact its generalization. In such a modeling architecture, the M0 layer is the real world as in [4] and [1]. It includes all possible things that we try to represent using the models residing at the M1 layer. That representation is more or less abstract and simplified, depending on how rich our models are. Models are defined using concepts defined in metamodels, so each metamodel determines how expressive its models can be. M2 is the layer where the metamodels are located. The metamodels are also defined using some concepts. A set of concepts used to define metamodels resides at the separate M3 layer at the top of this architecture and is called meta-metamodel. Meta-metamodel is nothing more than a metamodel that is conventionally elected to be used for other metamodels' definition; it also defines itself. The architecture is generalized to comprise not only models and metamodels based on an object-oriented meta-metamodel like MOF is, but also other systems, for instance: ontologies, Semantic Web technologies or non-technical representations.

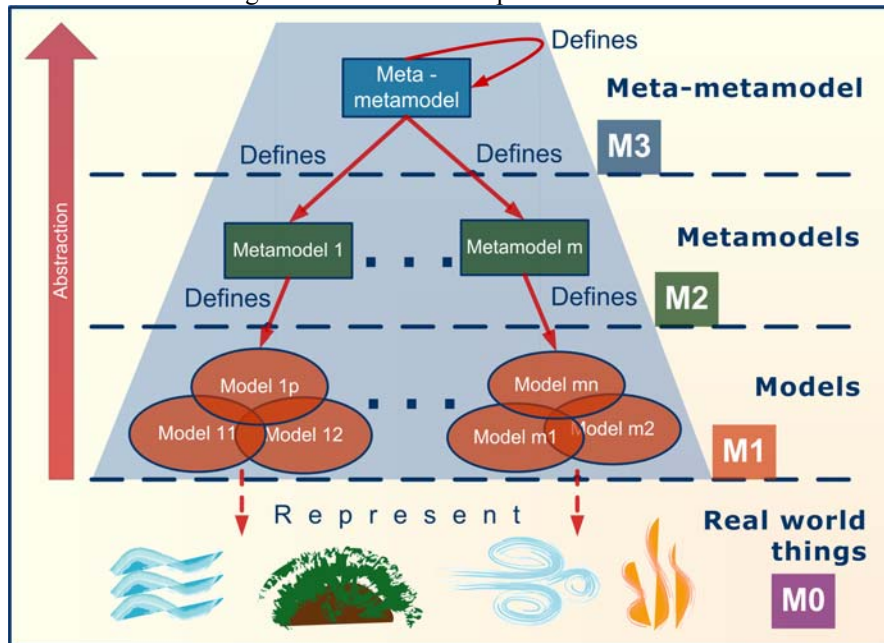


Fig. 3. General four-layer modeling architecture

This is a convenient place to introduce the concept of modeling spaces. A modeling space (MS) is a modeling architecture defined by a particular meta-metamodel. Metamodels defined by the meta-metamodel and models defined by those metamodels represent the real world from one point of view, i.e. from the point of view of that MS. As the meta-metamodel defines the core concepts used in defining all other metamodeling concepts, it is defined by itself. If it was defined by some

other concepts, it would not be a meta-metamodel, it would be an ordinary metamodel in some other MS.

Figure 4 shows a few examples of well-known MSs. The most straightforward example from this picture is the MOF MS. It is defined by the MOF meta-metamodel, which in turn is self-defined. It defines various metamodels, for instance Unified Modeling Language [14] or Ontology Definition Metamodel [6], that are used to describe models that represent things from the real world. The same reality is described in the context of other MSs, like RDF(S) or EBNF spaces. Many software engineers would associate the terms like model and modeling exclusively with UML aristocracy, taking EBNF-based models (Java, C#, C++ code) as more technical, flattened artifacts and ignoble citizens. However, Java (or C++, or some other) code is a model, since it represents some simplified abstraction of reality. The same is with XML code, databases, books, etc – they are all models, but modeled in terms of different MSs, defined by different meta-metamodels.

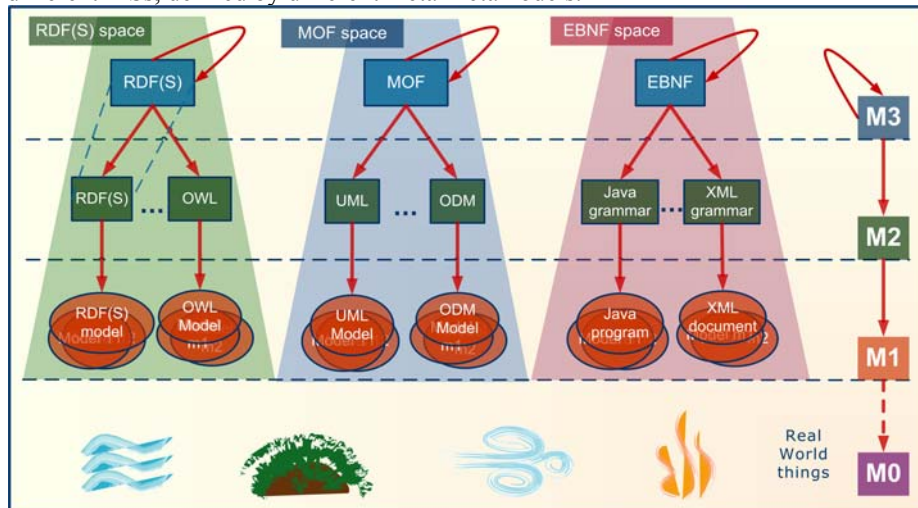


Fig. 4. RDF(S), MOF and EBNF modeling spaces

If we model the real world in a certain MS, we will use some models. If we model the same reality in another MS, we will describe it with different kinds of models, highlighting other characteristics when abstracting from reality. The models from the first MS will be a part of reality that we can model using the models from the second MS.

Figure 5 clarifies this duality by an example of the same thing being simultaneously a model and a real-world thing. Along the vertical axis, the world is modeled in the MOF MS. Along the horizontal axis is the EBNF space hierarchy, which is a real-world thing in the MOF space. An interesting observation here is that any MS, like the EBNF space or even the MOF space itself, is a part of the real world from the MOF-space point of view. In general, the way we model some business system or another “real” domain is pretty much the same as the way we model meta-metamodels, metamodels or models from another MS. Of course, these models

involve a certain level of abstraction, so there is a possibility of losing some information.

For many software engineers, this duality is complicated to understand at first. The fact that M1-M3 layers are fiction and above the M0 layer does not mean that meta-metamodels, metamodels and models are things outside of reality. Everything is in the real world; we just use a convention to put some things in layers, depending on the context.

MSs can be defined in more or less abstract manner. Some MSs are focused on conceptual (abstract or semantic) things, like models, ontologies, mathematical logics, etc. They are not interested in techniques for representation or sharing their abstractions. We call them conceptual MSs. However, we must have some techniques to materialize (or serialize) those MSs. We can do this using concrete MSs, which are equipped with syntax. Examples of those materializations are some syntax or databases.

Being able to represent bare syntax, concrete MSs need a means to express the semantics, i.e. the meaning of the data they carry. Conceptual MSs, on the other hand, are able to represent semantics, but need a means to represent their information physically. It is obvious that they should complement each other's representation abilities to create models that have both semantics and syntax. One of the most interesting examples of this symbiosis of various MSs can be found in OMG Model Driven Architecture. An example of a conceptual space is MOF MS, while an example of concrete MS is EBNF MS.

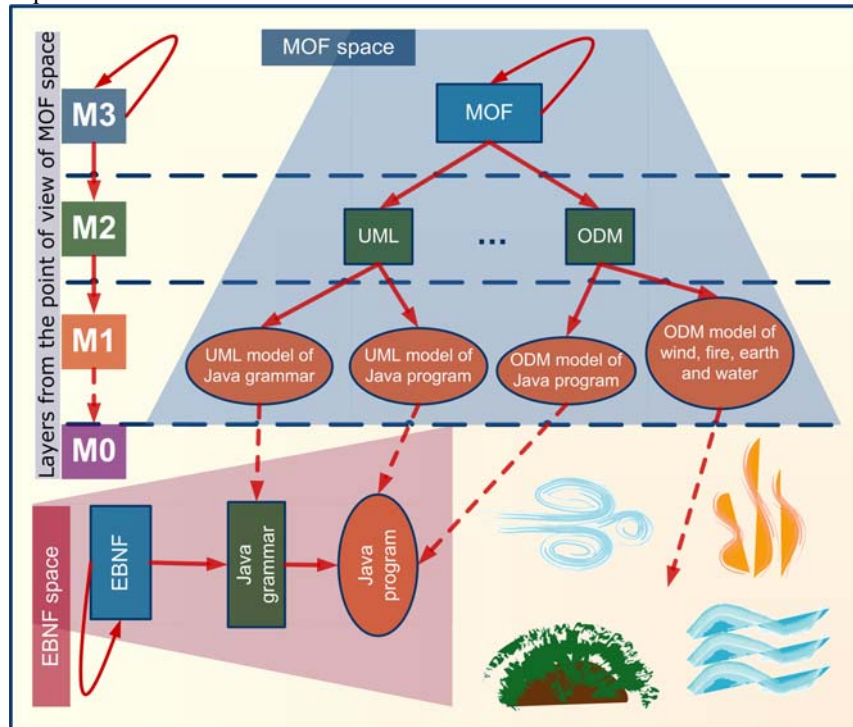


Fig. 5. MOF MS sees EBNF MS as a set of things from the real world

There are two types of usage scenarios for different MSs:

1. *Parallel spaces* – one MS models the same set of real-world things as another MS, but in another way. In this case, the relation between these MSs is oriented towards pure transformation, bridging from one space to another. Examples of such parallel MSs are MOF and RDF(S) MSs.
2. *Orthogonal spaces* – some MS models concepts from another MS, taking them as real-world things, i.e. one MS is represented in another MS. This relation is often used in round-trip engineering to facilitate different stages of modeling some system. For example, in order to make a Java program we could first use Java UML profile to create classes and method bodies, then transform this UML model into Java code, and complete the Java program using a Java IDE. Orthogonal MSs are also used when a conceptual MS is implemented using a certain concrete MS – for example, when one develops a MOF-based repository to run in a Java virtual machine.

4 The Touch of RDF(S) and MOF modeling spaces

Usage scenarios for parallel spaces most often pertain to conceptual MSs that model the same reality using different concepts. Each of these MSs is implemented in some other, more concrete MS, as a represented reality. In order to exchange models between conceptual MSs, it is necessary to provide transformations from one space to another. These transformations are also models [5], and should be developed in a MS that can represent both the source and the target MSs. Moreover, the transformation also has to be represented in some concrete MS orthogonal to the source and the target MSs, that leads the conceptual model of transformation to its implementation.

Figure 6 shows two parallel conceptual MSs, RDF(S) MS and MOF MS, and the space that represents them orthogonally, EBNF MS. MOF and RDF(S) model the real world in parallel, using some modeling languages (UML, ODM, OWL or other) that are defined using different meta-meta concepts. At the conceptual level, we could establish a transformation from one language to another, e.g. UML to ODM and vice versa, in the same MS. An example of a transformation modeling language for such purposes in MOF is Query-View-Transformation (QVT) [MOF QVT 2002]. RDF and RDF Schema, and three different dialects of OWL – OWL Full, OWL DL and OWL Lite are examples of languages at M2 layer of RDF(S) MS. Efforts to develop query and transformation language in RDF(S) MS are underway: Triple, RQL etc. [10].

We can also establish a transformation between MSs, a bridge that transforms RDF(S) concepts to MOF concepts at M3 layer. Using that bridge, we can transform any metamodel (language) defined in RDF(S) into its corresponding metamodel in defined in MOF (both are at M2 layer). Of course, we could expect an information loss depending on how meta-meta concepts (M3) are similar. RDF(S) concepts, `rdfs:Class` and `rdf:Property` are similar, but not the same as MOF Class, Association and Attribute.

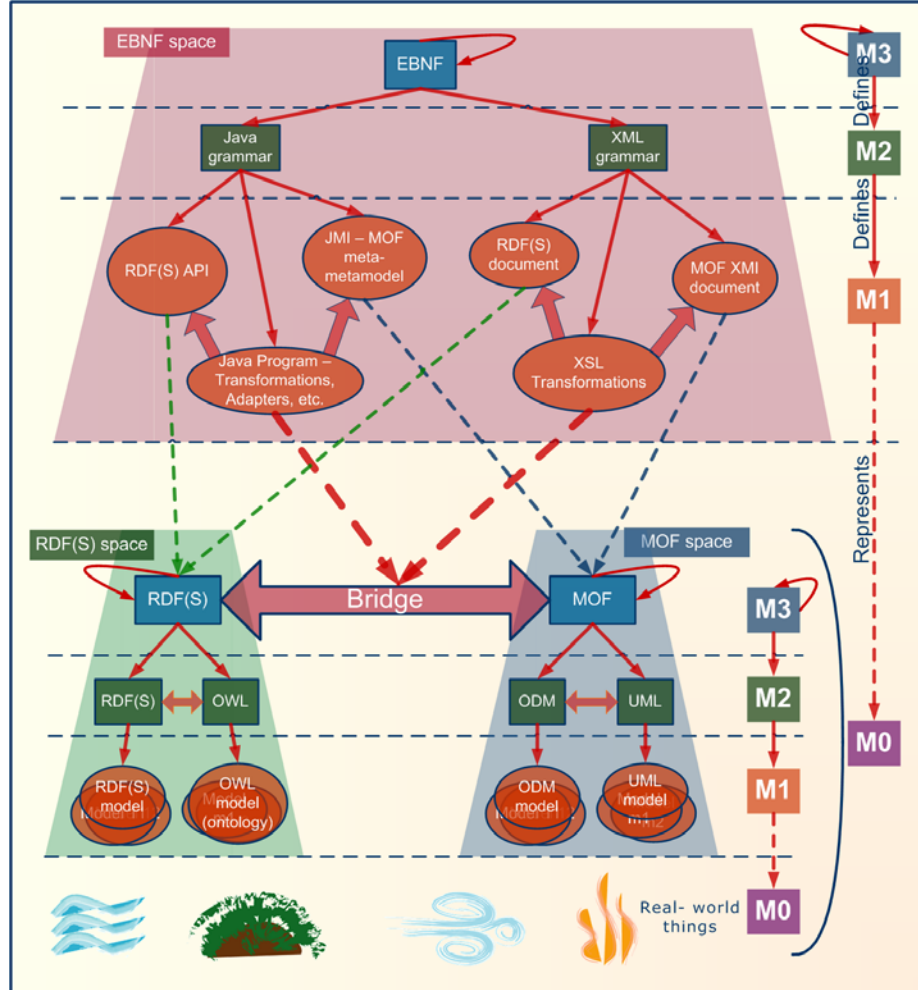


Fig. 6. Transformations between RDF(S) MS and MOF MS

Both MOF and RDF(S) spaces can be represented in other, more concrete MSs. They can be implemented using repositories, serialized into XML etc., which involves many MSs. For the sake of simplicity, we have skipped a few steps and have shown them as Java program codes and XML documents in the EBNF space. Models from the MOF space are modeled in Java code according to JMI standard, and in XML according to the MOF XMI. For languages from RDF(S), XML is the natural way of representation. They can also be modeled using Java APIs (Jena etc.).

As the RDF(S)-MOF bridge is also a model; it can be also represented in a concrete MS representing meta-metamodels that should be bridged. Examples include an XSLT that transforms a MOF XMI document into RDF(S) XML document and vice versa, a set of Java classes that adapt JMI interfaces to RDF(S) API interfaces, or

a Java program that does a batch transformation from a JMI-based code to an RDF(S) API-based one.

As Figure 6 shows explicitly, a single bridge models a transformation between two MSs at layer M3, between RDF(S) and MOF meta-metamodels. Transformations between metamodels situated in a single MS at M2 layer are internal to that MS. However, they can be implemented through some concrete MSs (e.g. EBNF for XSLT).

5 The Touch of the Semantic Web and MDA technical spaces

MS is a concept inspired by the concept of TS, which is defined as a working context with a set of additional concepts, body of knowledge, tools, required skills, and possibilities [11]. Fortunately, we can use MS to enhance this fuzzy definition of TS.

A technical space (TS) is a working context that includes various related MSs. Most often the TS is built around some MS, whereas the role of other MSs is supportive (e.g., implementation), or implicit (literature, know-how). For example, the MOF MS is at the center of the MDA TS. However, the MDA TS also partially includes other MSs: XML and EBNF in the area of XMI representation, EBNF in the area of repository implementation (JMI), an implicit MS that includes literature, etc. Transformations, for example to plain Java, C++ or VB code, are also models belonging to one or several MSs that are partially included in the MDA TS.

Figure 7 shows overlapping between the MDA TS and the Semantic Web TS in some MSs (most MSs belonging to these TSs are omitted for the sake of simplicity). The MDA TS is built around the MOF MS, which resides completely in the MDA TS. The MDA TS also includes OWL ontologies that model MOF-based concepts or contain some knowledge about MDA, which are parts of the RDF(S) MS. On the other hand, SW TS includes RDF(S) MS. Additionally, it includes parts of the MOF MS related to ODM metamodel and Ontology UML Profile, and two-way transformations from these MOF-based models to OWL. These transformations are also a part of the MDA TS. Recall that those transformations are also modeled, so they belong to some MSs as well. Some researches are trying to identify a way to enable transformations between different MSs at the M3 layer using just one two-way transformation for all three layers [4].

It follows from the above discussion that a TS includes one or more MSs and that each MS is a part of one or more TSs, whereas TS is a means for grouping MSs that have something in common or simply need to interact. The bridge connecting two MSs is also a means for connecting surrounding TSs.

6 Conclusions

We have presented here a concept of modeling spaces and used it in the definition of technical spaces and the explanation of two significant MSs – RDF(S) (from SW TS) and MOF (from MDA TS). We showed how models from RDF(S) MS are viewed in

MOF MS and vice versa. The SW and MDA interoperability issues are also shown with the explanation how it can be acquired from the point of view of MSs using bridges between meta-meta models.

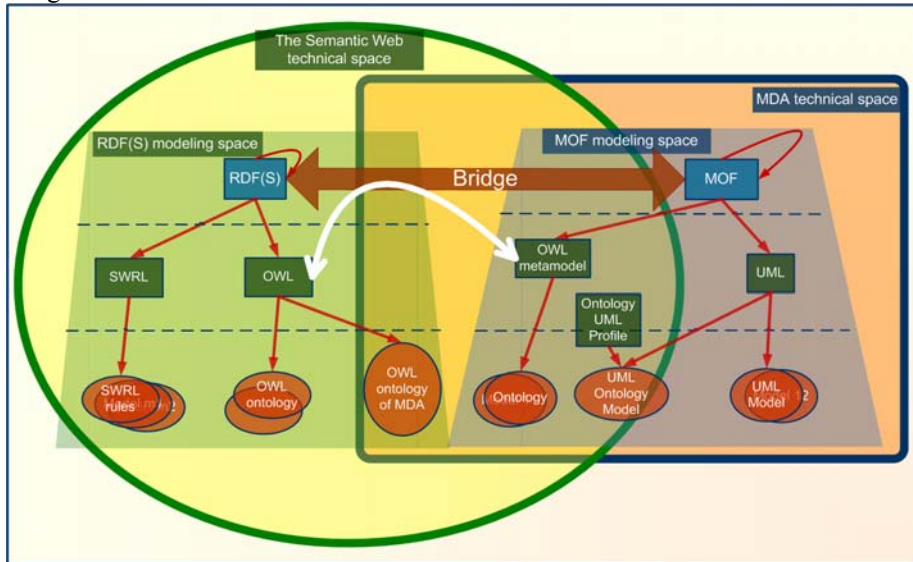


Fig. 7. The Semantic Web and Model Driven Architecture technical spaces

An interesting thing is that MSs are suitable to explain existing modeling architectures, even non-technical, and their implementations (MDA, Ontologies, EMF, EBNF, XML, etc.). Being in accordance with existing efforts, it is a suitable means for their clarification, explanation, and their conceptual connection.

References

1. Atkinson, C., Kühne, T., "Model-Driven Development: A Metamodeling Foundation" (Spec. issue on Model-Driven Development), IEEE Software, Vol. 20, No. 5, Sep/Oct, 2003, pp 36-41.
2. Berners-Lee, T., "Semantic Web Road Map", W3C Design Issues, <http://www.w3.org/DesignIssues/Semantic.html>, 1998.
3. Berners-Lee, T., Weaving the Web, Orion Business Books, London, 1999.
4. Bezivin, J., et al "A M3-Neutral infrastructure for bridging model engineering and ontology engineering," 1st International Conference on Interoperability of Enterprise Software and Applications, Geneva, Switzerland, 2005.
5. Bézivin, J., Dupé, G., Jouault, F., Pitette, G., Rougui, J.E., "First experiments with the ATL model transformation language: Transforming XSLT into XQuery," In Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of Model Driven Architecture, Anaheim, CA, USA, 2003.
6. Djuric, D., Gasevic, D., Devedzic, V., "Ontology Modeling and MDA", in Journal of Object Technology, vol. 4, no. 1, January-February 2005, pp. 109-128. http://www.jot.fm/issues/issue_2005_01/article3

7. Djuric, D., Gasevic, D., Devedzic, V., "Modeling Spaces", submitted to IEEE Software, 2005.
8. Gruber, T. R., "A translation approach to portable ontology specifications", Knowledge Acquisition, Vol. 5, No. 2, 1993.
9. Dean, M., Schrieber, G. (eds.), "OWL Web Ontology Language Reference", W3C Recommendation 10 February 2004 <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>
10. Hase, P., et. al., "A comparison of RDF query language proposals", Proc. Of The 3rd ISWC, Hiroshima, Japan, 7-11 November 2004. <http://xobjects.seu.edu.cn/resource/ISWC2004/pdf/32980502.pdf>
11. Kurtev, I., Bézivin, J., Aksit, M. "Technological Spaces: An Initial Appraisal", CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, 2002 <http://wwwhome.cs.utwente.nl/~kurtev/files/TechnologicalSpaces.pdf>
12. Miller, J., Mukerji, J. (eds.), "MDA Guide Version 1.0.1", OMG Document: www.omg.org/docs/omg/03-06-01.pdf, June 2003.
13. OMG spec., "Meta Object Facility (MOF) 2.0 Core Specification version 2.0 Final Adopted Specification" <http://www.omg.org/cgi-bin/apps/doc?ptc/03-10-04.pdf>, October 2003.
14. OMG spec., "Unified Modeling Language: Superstructure, version 2.0, Final Adopted Specification", <http://www.omg.org/cgi-bin/apps/doc?ptc/03-08-02.zip>, August 2003.
15. Hagget, P., Chorley, R. J. Models, Paradigms and New Geography, In Models in Geography, London, Methuen & Co., 1967.